AVSdeviceSDKWrapper RTC
環境構築・利用マニュアル

名城大学理工学部

メカトロニクス工学科

ロボットシステムデザイン研究室

目次

1. はじめに

1.1. 概要

AVSdeviceSDKWrapper RTCはAVS device SDKのプログラムとRTCを平行で動作させ，RTCからのトリガーをAVSdeviceSDKのプログラムに入力し，AVSdevice SDKのプログラムを操作し，OpenRTM上で音声対話を実現させる．

1.2. AVSdeviceSDK について

AVS device SDK は様々なデバイス製品にクラウド型音声対話エンジン Alexa を接続するための SDK であり，デバイスに音声制御の導入を実現することが可能となる．

AVDdeviceSDK の Alexa での位置は図 1 のようになる．

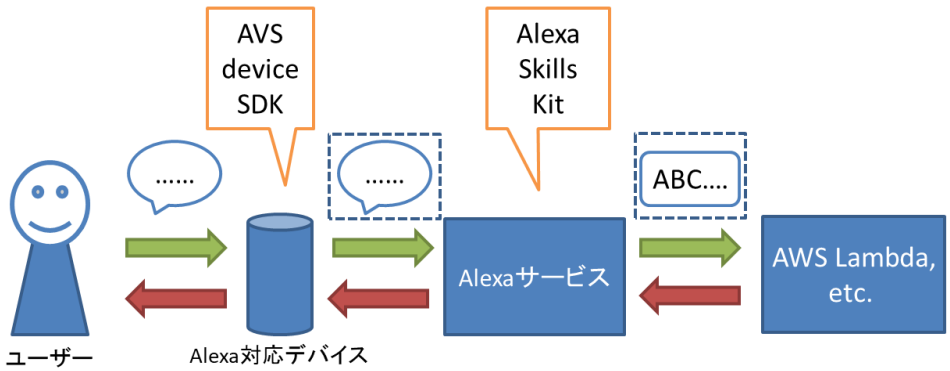

図 1　Alexa での AVSdeviceSDK の役割

図 1 より Alexa の動作の流れは以下のようになる．

1. ユーザーが AVSdeviceSDK 搭載の Alexa 対応デバイスに発話
2. デバイスはマイクで集音した音声データを，インターネット経由で Alexa サービス(Alexa Skills Kit)に送信
3. Alexa サービス(Alexa Skills Kit)は音声データを AI で解析し，AWS Lambda を呼び出して解析結果を伝える
4. Lambda はスキルの処理を実行し，その結果を Alexa サービス(Alexa Skills Kit)にレスポンスとして返す
5. Alexa サービス(Alexa Skills Kit)はレスポンスに応じた音声データを生成，デバイスに返信
6. デバイスは音声データを再生し，ユーザーに伝える

＊Alexa サービスのリクエスト先は，AWS Lambda のほかに任意の Web サービス設定できる．

以上より，AVSdeviceSDKはユーザーとAlexaサービスのやり取りを接続するためのSDKである．また，対話制御，音声合成，その他のデバイス操作などの機能を開発するためにはAlexaサービス(Alexa Skills Kit)を用いて開発する．

また，AVSdeviceSDKの主要コンポーネントは図2となる．

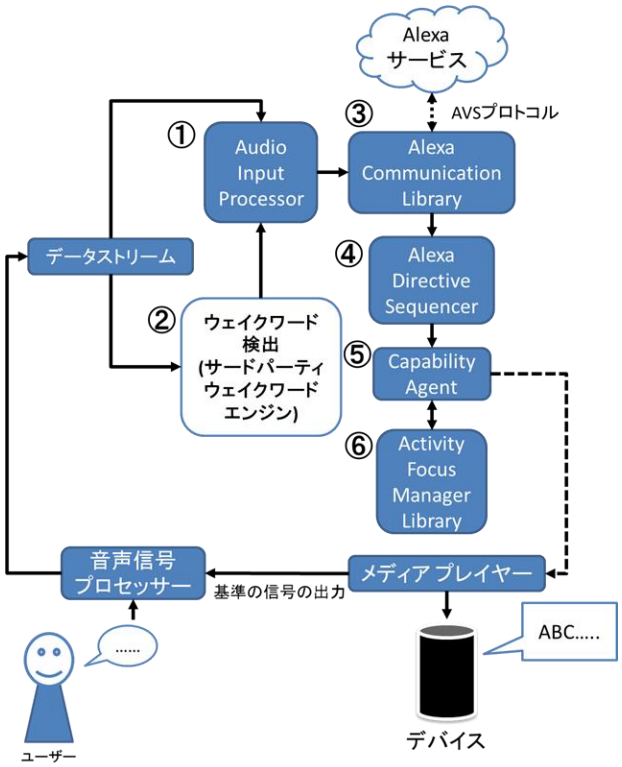

図 2　SDK の主要コンポーネント図

図2より主要コンポーネントの機能は以下になる．

① 音声入力プロセッサー

デバイスに搭載されているマイク，リモートマイク，その他の音声入力ソースからAVSdeviceSDKに送られてくる音声入力を処理する．

② ウェイクワード検出

入力ストリームからAlexaウェイクワードを検出する独自のウェイクワードエンジンを追加するためのフックが備わっている．

③ Alexa Communicationsライブラリ

デバイスとAlexaサービスとの主要なコミュニケーションチャネルとして機能する．

④ Alexa Directive Sequencerライブラリ

Alexa Voice Service (AVS) からの ディレクティブ の順序を管理する．

⑤ 機能エージェント

Alexaを中心とした，ディレクティブとイベントの対話操作を扱う．それぞれの機能エージェントは，AVS APIによって公開される特定のインターフェースに対応する．

⑥ Activity Focus Managerライブラリ

AVS対話モデルの仕様に従い，チャネルの入力と出力に優先順位を付ける．

1.3. RTC の機能と動作

AVSdeviceSDKのフレームワークの制約を考慮に入れ，AVSdeviceSDKを外部的に動作させるRTCを開発した．このRTCはデータポートからの特定の入力に応じてAVSdeviceSDKのプログラムの起動や再起動，終了などを行うことができる．使用例としては，カメラコンポーネントが人を認識したときにデータポートへ特定の入力を行うことにより自動で対話を開始することができる．この機能と動作イメージを図3に示す．

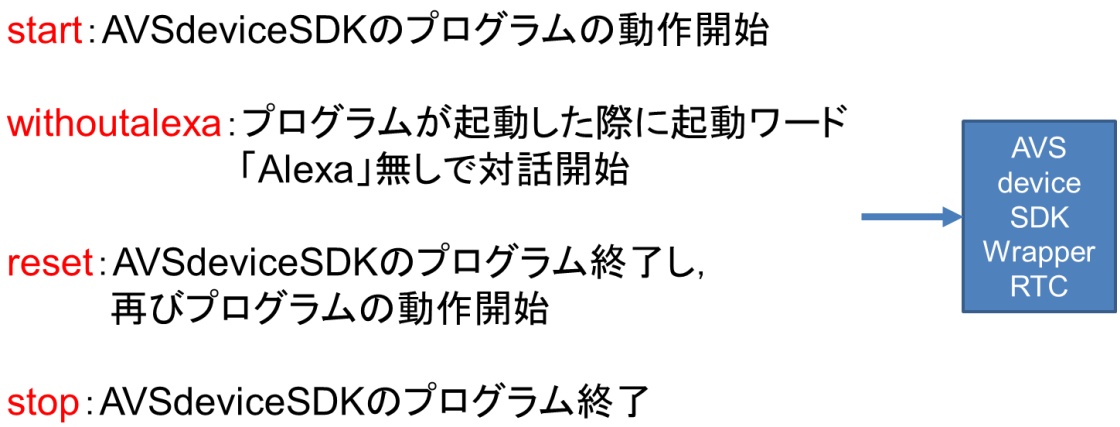

図 3　RTC の機能と動作

図3よりデータポートから”start”が入力された場合，AVSdeviceSDKWrapper RTCはAVSdeviceSDKのプログラムを動作させる．AVSdeviceSDKのプログラムが起動した際に，”withoutalexa”が入力された場合，AVSdeviceSDKWrapper RTCは起動ワード「Alexa」なしで対話を開始する．”reset”が入力された場合，AVSdeviceSDKWrapper RTCは一度AVSdeviceSDKのプログラムを終了させ，再びAVSdeviceSDKのプログラムを動作させる．”stop”が入力された場合，AVSdeviceSDKWrapper RTCはAVSdeviceSDKのプログラムを終了させる．

1.4. AVSdeviceSDK のライセンス

AVSdeviceSDKWrapper RTC を動作させるために alexa 様から提供されているライブラリを利用しています．そちらのライセンスに関しては以下のファイルの指示に従ってください．

URL: https://github.com/alexa/avs-device-sdk/blob/master/LICENSE.txt

1.5. 開発環境

動作環境を以下に示す.

表 1　動作 PC 環境

PC	Panasonic Let’s Note(CF-LX3)
OS	Ubuntu16.04 LTSのみ動作

表 2　ソフトウェア環境

ソフトウェア名	バージョン
OpenRTM-aist C++	1.1.2-RELEASE
C++	11 or later
GNU Compiler Collection (GCC)	4.8.5 or later
Cmake	3.1 or later
Libcurl	7.50.2 or later
nghttp2	1.0 or later
OpenSSL	1.0.2 or later
Doxygen	1.8.13 or later
SQLite	3.19.3 or later
Gstreamer	1.10.4 or later
GStreamer Base Plugins	1.10.4 or later
GStreamer Good Plugins	1.10.4 or later
GStreamer Libav Plugin	1.10.4 or later
GStreamer Ugly Plugins	1.10.4 or later
GStreamer Bad Plugins	1.10.4 or later
PortAudio	v190600_20161030
Crypto Library	
Libsoup	
libfaad-dev	
g++, gcc	4.8 and 5

2. AVSdeviceSDKWrapper RTC の仕様

AVSdeviceSDKWrapper RTC の仕様の詳細を示す.

表 3　AvsDeviceSDKWrapper RTC の仕様

RTC の名称			
AvsDeviceSDKWrapper			
入力ポート			
名称	データ型	データ	
Orderin	TimedString	start の入力に対し，プログラムの起動を行う restart の入力に対し，起動ワード無しで対話開始 reset の入力に対し，再起動を行う stop の入力に対し，強制終了	
主なコンフィグレーションパラメータ			
名称	データ型	デフォルト値	説明
execution_path	string	/execution/path	実行ファイルのパスの指定

3. AVSdeviceSDK の導入

3.1. AVS の接続するための ID 登録

AVS の接続するための ID 登録の手順を以下に示す.

1. Amazon developer サイトへログイン
2. ALEXA のタブをクリック
3. Alexa Voice Service Get Started をクリック
4. CREATE PRODUCT をクリック
5. https://github.com/alexa/alexa-avs-sample-app/wiki/Create-Security-Profile を参考にして ID 登録

3.2. AVSdeviceSDK をインストールするフォルダーの作成

以下のコマンドを実行する.

```
$cd ~/<AVSdeviceSDK を保存する場所>
$mkdir sdk-folder && cd sdk-folder
$mkdir sdk-build sdk-source third-party application-necessities
$cd application-necessities && mkdir sound-files
```

3.3. AVSdeviceSDK の必要なソフトウェアインストール

3.3.1. マイクの録音ソフトウェアインストール

以下のコマンドを実行する.

```
$cd ~/<AVSdeviceSDK を保存する場所>/sdk-folder/third-party
$wget -c http://www.portaudio.com/archives/pa_stable_v190600_20161030.tgz
$tar zxf pa_stable_v190600_20161030.tgz
$cd portaudio
$./configure --without-jack
$make
```

3.3.2. AVS device SDK ソフトウェアインストール

以下のコマンドを実行する.

```
$cd ~/<AVSdeviceSDK を保存する場所>/sdk-folder/sdk-source
$sudo git clone https://github.com/alexa/avs-device-sdk.git
```

3.3.3. SDK で使用するワード検出ソフトウェアインストール

以下のコマンドを実行する.

```
$cd ~/<AVSdeviceSDK を保存する場所>/sdk-folder/third-party
$sudo git clone https://github.com/Kitt-AI/snowboy.git
```

3.3.4. 参照音声データのインストール

以下のコマンドを実行する.

```
$cd ~/ <AVSdeviceSDK を保存した場所>/sdk-folder/application-necessities/sound-files
$wget -c https://images-na.ssl-images-amazon.com/images/G/01/mobile-apps/dex/alexa/alexa-voice-service/docs/audio/states/med_system_alerts_melodic_02._TTH_.mp3
$wget -c https://images-na.ssl-images-amazon.com/images/G/01/mobile-apps/dex/alexa/alexa-voice-service/docs/audio/states/med_system_alerts_melodic_02_short._TTH_.wav
$wget -c https://images-na.ssl-images-amazon.com/images/G/01/mobile-apps/dex/alexa/alexa-voice-service/docs/audio/states/med_system_alerts_melodic_01._TTH_.mp3
$wget -c https://images-na.ssl-images-amazon.com/images/G/01/mobile-apps/dex/alexa/alexa-voice-service/docs/audio/states/med_system_alerts_melodic_01_short._TTH_.wav
```

3.4. ファイルの編集

コンポーネントを実行するにあたって複数のファイルを編集する必要がある．その手順を以下に示す.

以降，<AVSdeviceSDK を保存した場所>を AVSdeviceSDKDownload とする.

3.4.1. ID 情報をファイルに反映及び参照音データのパスの設定

```
$cd ~/AVSdeviceSDKDownload/sdk-folder/sdk-source/avs-device-sdk/Integration
$emacs AlexaClientConfig.json
```

編集するエディタとしては emacs を用いているが，その他のテキストエディタを用いても問題ない.

以下がファイルをすべて編集する内容

```
{
      "authDelegate":{
         "clientSecret":"<登録した ClientSecret のコード>",
         "deviceSerialNumber":"12345678(適当な番号で)",
         "refreshToken":" $ {SDK_CONFIG_REFRESH_TOKEN} ",
         "clientId":"<登録した ClientID のコード>",
         "productId":"<登録した ProductID のコード>"
       },
   "alertsCapabilityAgent":{
          "databaseFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessities/alerts
.db ",
          "alarmSoundFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessities/s
ound-files/med_system_alerts_melodic_01._TTH_.mp3 ",
         "alarmShortSoundFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessit
ies/sound-files/med_system_alerts_melodic_01_short._TTH_.wav ",
        "timerSoundFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessities/sou
nd-files/med_system_alerts_melodic_02._TTH_.mp3",
         "timerShortSoundFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessiti
es/sound-files/med_system_alerts_melodic_02_short._TTH_.wav "
     },
   "settings":{
          "databaseFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessities/setti
ngs.db ",
           "defaultAVSClientSettings":{
                "locale":"en-US”
             }
      }
    "certifiedSender":{
"databaseFilePath":"/AVSdeviceSDKDownload/sdk-folder/application-necessities/certifiedSend
er.db "
      }
  }
```

3.4.2. AVSdeviceSDK のプログラム変更

```
$cd ~/ AVSdeviceSDKDownload /sdk-folder/sdk-source/avs-device-sdk/SampleApp/src
$emacs main.cpp
```

編集するエディタとしては emacs を用いているが，その他のテキストエディタを用いても問題ない.

以下がファイルをすべて編集する内容

```
#include "SampleApp/SampleApplication.h"
#include <cstdlib>
#include <string>

int main() {
    std::string pathToConfig;
    std::string pathToInputFolder;
    std::string logLevel;
    auto sampleApplication =
alexaClientSDK::sampleApp::SampleApplication::create(pathToConfig="/AVSdeviceSDKDow
nload/sdk-folder/sdk-build/Integration/AlexaClientSDKConfig.json",pathToInputFolder="/AVS
deviceSDKDownload/sdk-folder/third-party/snowboy/resources",logLevel="DEBUG9");
    if (!sampleApplication) {
alexaClientSDK::sampleApp::ConsolePrinter::simplePrint("Failed to create to SampleApplicatio
n!");
        return EXIT_FAILURE;
    }
    sampleApplication->run();
    return EXIT_SUCCESS;
}
```

3.5. AVSdeviceSDK のコンパイル

g++と gcc のバージョンを両方とも 4.8 に変更する．バージョン切り替えは以下で行う．

```
$sudo update-alternatives --config gcc
$sudo update-alternatives --config g++
```

コンパイルを行うコマンドは以下になる．

<AVSdeviceSDK を保存した場所>を AVSdeviceSDKDownload とする．

```
$cd ~/ AVSdeviceSDKDownload /sdk-folder/sdk-build
$cmake /AVSdeviceSDKDownload/sdk-folder/sdk-source/avs-device-sdk/ -DKITTAI_KEY_WORD_DETECTOR=ON -DKITTAI_KEY_WORD_DETECTOR_LIB_PATH=/AVSdeviceSDKDownload/sdk-folder/third-party/snowboy/lib/ubuntu64/libsnowboy-detect.a -DKITTAI_KEY_WORD_DETECTOR_INCLUDE_DIR=/AVSdeviceSDKDownload/sdk-folder/third-party/snowboy/include -DPORTAUDIO=ON -DPORTAUDIO_LIB_PATH=/AVSdeviceSDKDownload/sdk-folder/third-party/portaudio/lib/.libs/libportaudio.a -DPORTAUDIO_INCLUDE_DIR=/AVSdeviceSDKDownload/sdk-folder/third-party/portaudio/include -DGSTREAMER_MEDIA_PLAYER=ON
$make
```

3.6. AVS の接続

AVS を接続するにあたり，以下のことを行う．

<AVSdeviceSDK を保存した場所>を AVSdeviceSDKDownload とする．

3.6.1. コマンド実行

```
$cd ~/ AVSdeviceSDKDownload /sdk-folder/sdk-build
$python AuthServer/AuthServer.py
```

3.6.2. AVS の接続するためのサイト表示

http://localhost:3000 をサイト表示し，Amazon developer サイトにログイン，接続完了表示をさせ，ウインドウを閉じる．

4. RTC の導入・動作

4.1. RTC のコンパイル

g++と gcc のバージョンを両方とも 5 に変更する．バージョン切り替えは以下で行う．

```
$sudo update-alternatives --config gcc
$sudo update-alternatives --config g++
```

コンパイルを行うコマンドは以下になる．

```
$cd ~/<RTC を保存した場所>/AVSdeviceSDKWrapper
$mkdir build && cd build
$cmake ..
$make
```

4.2. RTC の起動方法

RTC の起動方法は以下になる．

ターミナル 1:eclipse 起動

```
$eclipse
```

ターミナル 2：RTC の起動

```
$cd ~/<RTC を保存した場所>/AVSdeviceSDKWrapper/build/src
$./AvsDeviceSDKWrapperComp
```

4.3. RTC の利用方法

ターミナル 1 で起動した eclipse において，RTC の利用手順を以下に示す.

1. メニューバーから「ウィンドウ」→「パースペクティブを開く」→「RT System Editor」を選択する.
2. 「Open New System Editor」をクリックし，システムエディタを立ち上げる.
3. ネームサーバーに登録されている RT コンポーネントをシステムエディタ上にドラック&ドロップする.
4. AvsDeviceSDKWrapper をクリックし，ConfigurationView において，AVSdeviceSDK の SampleApp の実行ファイルのパスを確認し，適宜環境に合わせたパスを指定し，「適用」ボタンを押す.（パスの例）/ AVSdeviceSDK を保存した場所/sdk-folder/sdk-build/SampleApp/src/SampleApp
5. 「All Activate」をクリックする.
6. Execute 状態になると，データポートからの特定の入力に応じて AVSdeviceSDK のプログラムの起動や再起動，終了などを行う.
7. 「All Deactivate」をクリックすると会話を終了する.

4.4. RTC の動作説明

RTC のデータポートに”start”を入力すると RTC を実行したターミナルは図 4，5 になる．

```
Running app with log level: DEBUG9
2017-12-07 05:20:48.850 [  1] W Logger:debugLogLevelSpecifiedWhenDebugLogsCompiledOut:
level=DEBUG9:

WARNING: By default DEBUG logs are compiled out of RELEASE builds.
Rebuld with the cmake parameter -DCMAKE_BUILD_TYPE=DEBUG to enable debug logs.

2017-12-07 05:20:48.850 [  1] I ConfigurationNode:initializeSuccess
2017-12-07 05:20:48.906 [  1] I AlertScheduler:executeScheduleNextAlertForRendering::n
o work to do.
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=SpeechSynthesizer,name=Speak,handler=0x2283bd0,policy=BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=AudioPlayer,name=ClearQueue,handler=0x22848b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=AudioPlayer,name=Stop,handler=0x22848b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=AudioPlayer,name=Play,handler=0x22848b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=SpeechRecognizer,name=ExpectSpeech,handler=0x2282970,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=SpeechRecognizer,name=StopCapture,handler=0x2282970,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=Alerts,name=DeleteAlert,handler=0x22852c0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=Alerts,name=SetAlert,handler=0x22852c0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=System,name=SetEndpoint,handler=0x22ae480,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=System,name=ResetUserInactivity,handler=0x2282598,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=Speaker,name=SetMute,handler=0x22af0b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=Speaker,name=AdjustVolume,handler=0x22af0b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=Speaker,name=SetVolume,handler=0x22af0b0,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=TemplateRuntime,name=RenderPlayerInfo,handler=0x22adb90,policy=NON_BLOCKING
2017-12-07 05:20:48.908 [  1] I DirectiveRouter:addDirectiveHandlers:action=added,name
space=TemplateRuntime,name=RenderTemplate,handler=0x22adb90,policy=NON_BLOCKING
##############################
#       Connecting...        #
##############################

########################################
#       Alexa is currently idle!       #
########################################
```

図 4　データポートに”start”を入力したときのターミナルの状態 1

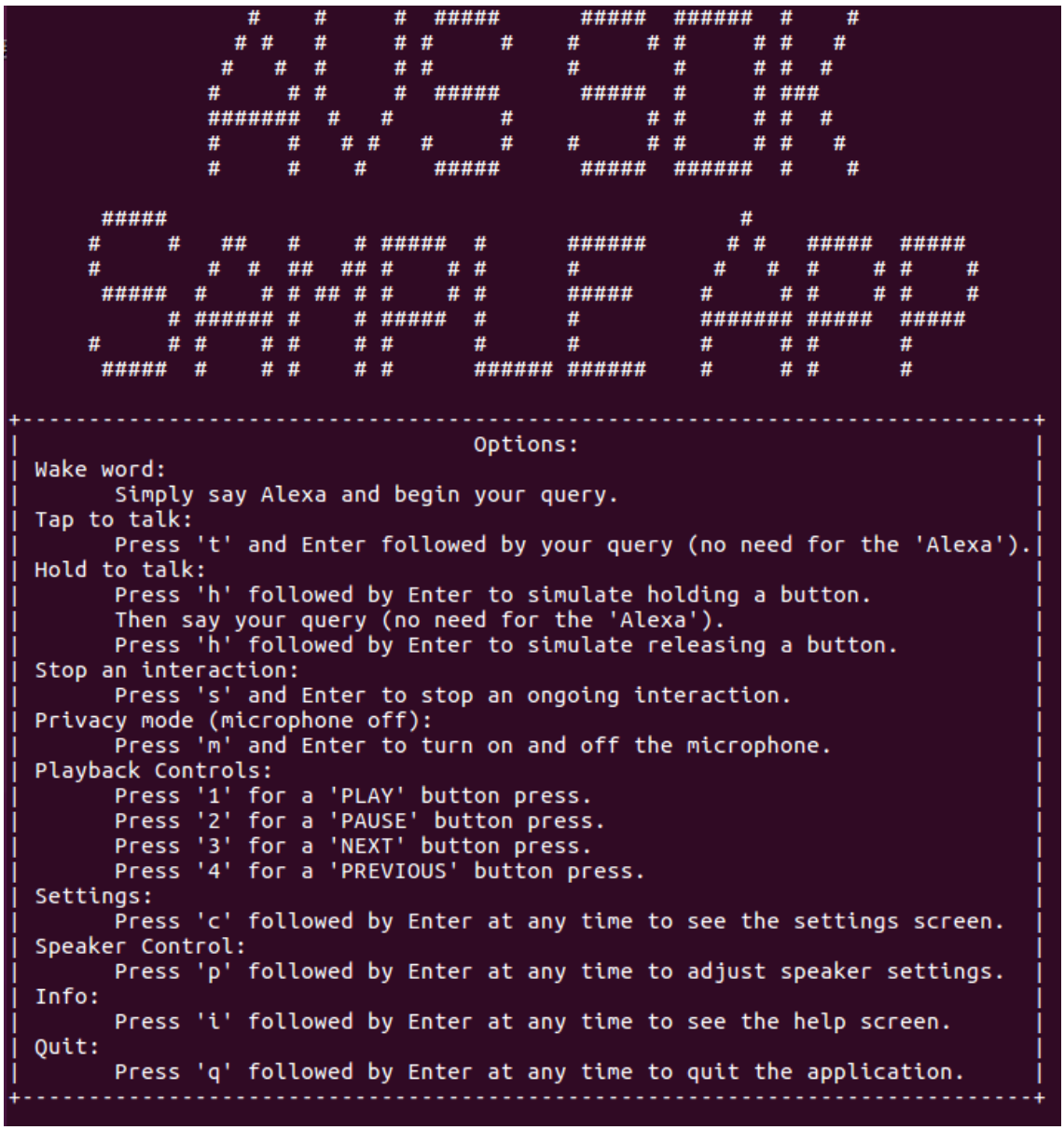

図 5　データポートに"start"を入力したときのターミナルの状態 2

図 4，5 の状態は AVSdeviceSDK の動作評価用サンプルプログラムが起動した状態となっている．図 4 ではデバックログを出力しつつ，SDK の初期化を行っている．「Connecting…」ではクラウド型音声対話エンジン Alexa の接続を示している．「Alexa is currently idle!」では Alexa からの接続が完了し，利用可能であることを示している．図 5 ではサンプルプログラムのオプション設定についての説明であり，標準入力によって SDK の設定とサンプルプログラムの動作をユーザーから変更を行うことができる．また，この状態ではマイクからの音声入力待ち状態となっており，起動ワード「Alexa」と発言することで対話が開始される．

対話が開始された状態が図 6 となる.

```
############################
#       Listening...       #
############################

2017-12-07 05:21:00.682 [  2] I DirectiveProcessor:setDialogRequestIdLocked:oldValue=,
newValue=ca446827-c28a-4ea0-9934-d4985af3a99e
2017-12-07 05:21:03.120 [  3] I DirectiveSequencer:onDirective:directive={"namespace\:
"SpeechRecognizer"\,name\:"StopCapture"\,messageId\:"75a8afdb-f2bd-4f10-b5d5-28340b266
3bf"\,dialogRequestId\:""}
2017-12-07 05:21:03.120 [  4] I DirectiveRouter:preHandleDirective:messageId=75a8afdb-
f2bd-4f10-b5d5-28340b2663bf,action=calling
2017-12-07 05:21:03.120 [  5] I DirectiveRouter:handleDirective:messageId=75a8afdb-f2b
d-4f10-b5d5-28340b2663bf,action=calling
############################
#       Thinking...        #
############################

2017-12-07 05:21:03.130 [  6] I InProcessAttachmentReader:readFailed:reason=SDS is clo
sed
2017-12-07 05:21:03.702 [  3] I DirectiveSequencer:onDirective:directive={"namespace\:
"SpeechSynthesizer"\,name\:"Speak"\,messageId\:"b2aca208-4198-433e-8446-cf22eb1c9a64"\
,dialogRequestId\:"ca446827-c28a-4ea0-9934-d4985af3a99e"}
2017-12-07 05:21:03.702 [  4] I DirectiveRouter:preHandleDirective:messageId=b2aca208-
4198-433e-8446-cf22eb1c9a64,action=calling
2017-12-07 05:21:03.702 [  5] I DirectiveRouter:handleDirective:messageId=b2aca208-419
8-433e-8446-cf22eb1c9a64,action=calling
########################################
#       Alexa is currently idle!       #
########################################

2017-12-07 05:21:03.910 [  7] I InProcessAttachmentReader:readFailed:reason=SDS is clo
sed
############################
#       Speaking...        #
############################

########################################
#       Alexa is currently idle!       #
########################################
```

図 6 対話状態のときのターミナルの状態

図 6 の状態では「Listening…」でユーザーからの用件の発話の入力待ち状態を示す.「Thinking...」でその用件の発話を音声データ化し，その音声データを Alexa に送信している.「Alexa is currently idle!」で Alexa が対話処理を行った音声データを受け取り，Alexa からの接続が完了する.「Speaking...」でその音声データを出力し，発話を行う．次の「Alexa is currently idle!」では Alexa からの接続が完了し，利用可能であることを示し，再び，マイクからの音声入力待ち状態となる.

RTC を Deactivate 状態になると RTC を実行したターミナルは図 7 になる．

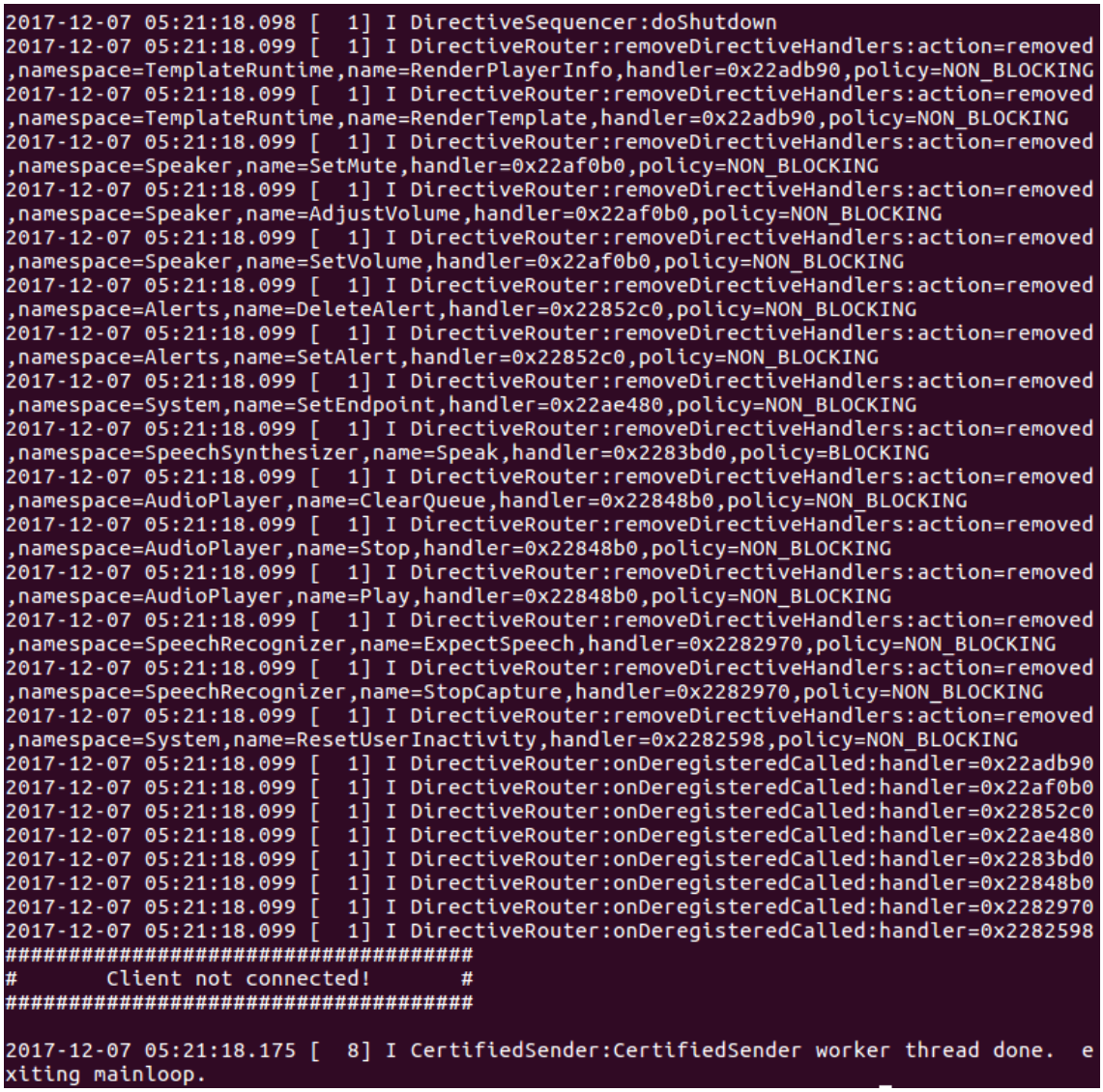

図 7 Deactivate 状態のときのターミナルの状態

図 7 の状態では Alexa の接続を切り，マイクからの入力を OFF 状態にし，SDK のプログラムを終了させ，「Client not connected!」を表示し，対話を終了する．

データポートからの入力に対しては，プログラムが動作した際に，「withoutalexa」の入力があった場合，図 6 の状態になり，起動ワード「Alexa」無しで対話が開始される．
「reset」の入力があった場合，図 4，5 の状態になり，プログラムの再起動を行う．
「stop」の入力があった場合，図 6 の状態になり，強制終了する．

5. 注意・補足事項

- g++と gcc のバージョン変更なしでコンパイルを行った場合，エラーが起こるので注意してください．(AVSdeviceSDK の起動ワード機能と OpenRTM の g++と gcc のバージョンの違いによるエラー)
- AVSdeviceSDK の動作評価用サンプルプログラムは最低限の対話(例:What time is it now?—Nine o'clock)しか行いません．対話制御や音声合成，その他のデバイス操作などの設計は Alexa Skills Kit を用いて開発を行います．こちらの URL を参考に開発を行ってください．

  URL: https://developer.amazon.com/ja/alexa-skills-kit

  URL:https://developer.amazon.com/ja/blogs/alexa/post/6e716e5c-55b0-445b-b936-9cfac4712e7b/training-1
- AVSdeviceSDKWrapper RTC のライセンスに関しては以下のファイルの指示に従ってください．

  URL: https://github.com/rsdlab/AVSdeviceSDKWrapperRTC/blob/master/LICENSE

6. 参考資料

AVS Device SDKの公式ホームページ

URL:https://developer.amazon.com/ja/alexa-voice-service/sdk

AVS Device SDKのGithubページ

URL:https://github.com/alexa/avs-device-sdk

音声対話エンジンAlexaの公式ホームページ

URL:https://developer.amazon.com/ja/alexa

起床ワード認識エンジンsnowboyのGithub

URL:https://github.com/Kitt-AI/snowboy.git

Alexa Skills Kitの公式ホームページ

URL: https://developer.amazon.com/ja/alexa-skills-kit

Alexaスキル開発トレーニングシリーズ 第1回 初めてのスキル開発

URL: https://developer.amazon.com/ja/blogs/alexa/post/6e716e5c-55b0-445b-b936-9cfac4712e7b/training-1